

Question 1

Correct

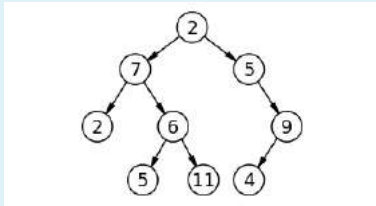
Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Tree adalah sebuah struktur data yang terdiri atas *node* yang terhubung satu sama lain sedemikian rupa sehingga menyerupai sebuah pohon fisik. Sebuah *tree* memiliki sebuah *root node*, dan setiap *node* pada *tree* memiliki 0 atau lebih *children nodes*, dengan batasan bahwa tidak ada hubungan (*edge*) antar *node* yang terduplikasi, dan tidak hubungan yang mengarah balik ke *root node*.

Struktur sebuah *tree* dapat dimodelkan secara rekursif, dimana setiap *node* adalah juga sebuah *tree* (atau *subtree*). Dengan menggunakan pendekatan rekursif ini, sebuah *tree* adalah sebuah *node*, dimana setiap *node* terdiri atas sebuah nilai data dan sekumpulan *node* yang merupakan anak (*children*) dari *node* tersebut (*list of children nodes* alias *list of subtrees*). Sebuah *tree* dikatakan sebagai *binary tree* manakala setiap *node* memiliki paling banyak 2 (dua) anak.

Sebuah struktur data *tree* dapat dibuat generik sehingga tipe dari nilai data pada setiap *node* dapat disesuaikan menurut kebutuhan aplikasi. Gambar berikut menunjukkan contoh sebuah *binary tree* dengan tipe data *integer*.



Buatlah sebuah kelas *BinaryTree* dengan menggunakan kelas generik dalam bahasa C++. Kelas memiliki beberapa fungsi berikut ini:

1. *Default constructor* dan *copy constructor*
2. Destruktor
3. Setter yang mengubah nilai data pada *node*.
4. Method yang menginisiasi dan menambahkan sebuah *child node* baru dengan sebuah nilai *value* dan mengembalikan *node (subtree)* yang baru ditambahkan. Gunakanlah *exception* untuk membatasi jumlah anak paling banyak 2

Petunjuk:

- *List of children* diimplementasikan dengan menggunakan *vector* dari STL C++.

Lengkapi file [BinaryTree.hpp berikut](#) dan lumpulkan implementasi kembali dalam file **BinaryTree.hpp**

Update:

- Ditambahkan using namespace std di file BinaryTree.hpp
- method addChild diubah, awalnya return `BinaryTree<T>*` sekarang menjadi `BinaryTree<T>&`

C++11

[BinaryTree.hpp](#)

Score: 150

Blackbox

Score: 150

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	10	Accepted	0.00 sec, 2.89 MB
2	10	Accepted	0.00 sec, 2.91 MB
3	10	Accepted	0.00 sec, 3.04 MB
4	10	Accepted	0.00 sec, 3.00 MB

No	Score	Verdict	Description
5	10	Accepted	0.00 sec, 2.94 MB
6	10	Accepted	0.00 sec, 2.91 MB
7	10	Accepted	0.00 sec, 2.90 MB
8	10	Accepted	0.00 sec, 2.89 MB
9	10	Accepted	0.00 sec, 3.09 MB
10	10	Accepted	0.00 sec, 2.88 MB
11	10	Accepted	0.00 sec, 3.04 MB
12	10	Accepted	0.00 sec, 3.09 MB
13	10	Accepted	0.00 sec, 2.97 MB
14	10	Accepted	0.00 sec, 2.93 MB
15	10	Accepted	0.00 sec, 2.96 MB

Time limit	1 s
Memory limit	64 MB

Buatlah sebuah file **cpp** yang berisi implementasi **fungsi generik**. Berikut kebutuhan dari fungsi generik pada praktikum ini.

1. Buatlah fungsi **func(param1, param2)** yang dapat menerima **dua** parameter input bertipe apapun dengan ketentuan: param1 dan param2 **bertipe sama**, dan param1 dan param2 **boleh bertipe beda**.
Jika param1 dan param2 bertipe sama, fungsi ini akan menghasilkan keluaran "**Masukan Anda: <param1> dan <param2>, memiliki tipe yang sama**" diakhiri dengan **newline**.
Jika param1 dan param2 mungkin bertipe beda, fungsi ini akan menghasilkan keluaran "**Masukan Anda: <param1> dan <param2>, mungkin memiliki tipe yang berbeda**" diakhiri dengan **newline**.
2. Terdapat **kasus khusus** untuk **point 1**, yaitu ketika **param1** bertipe **char** dan **param2** bertipe **int**, fungsi ini akan mengeluarkan nilai **param1** sebanyak nilai **param2**.

Contoh:

```
func<char,int>('a', 2)
```

akan menghasilkan **keluaran**:

```
a
a
```

HINT:

Berikut merupakan fitur template tambahan pada c++ yang dapat digunakan pada praktikum ini.

Template Specialization


Digunakan untuk membuat perlakuan khusus terhadap suatu class atau tipe tertentu.

```
// Menghasilkan t1 jika t1>=t2
template<class T>
T greaterOrEqual(T t1,T t2) {
    return t1 >= t2 ? t1 : t2;
}

// Template specialization untuk tipe string
// Jika parameter template bertipe string maka
// fungsi greaterOrEqual dibawah yang akan dipanggil
// Menghasilkan t1 jika jumlah huruf t1 >= jumlah huruf t2
template<>
string greaterOrEqual<string>(string t1,string t2) {
    return t1.size() >= t2.size() ? t1 : t2;
}
```

Kumpulkan satu file bernama **Generik.cpp**.

C++14

 [Generik.cpp](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	25	Accepted	0.00 sec, 3.00 MB
2	25	Accepted	0.00 sec, 2.97 MB
3	25	Accepted	0.01 sec, 2.96 MB
4	25	Accepted	0.00 sec, 2.93 MB

Question 3

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Mr. Joe bekerja sebagai seorang sekretaris yang akan mencatat memo-memo yang ditujukan untuk orang di himpunannya dan menyimpan memo tersebut. Dalam menulis memo, Mr. Joe memerlukan kertas, tinta, dan energi yang sudah sangat terkuras karena banyaknya tubes yang harus dikerjakannya. Kertas dan tinta dapat habis dan Mr. Joe tinggal meminta lagi pada bendahara. Energi juga dapat dipulihkan dengan istirahat (tetapi tetap dengan jumlah yang sedikit karena tubes selalu menghantui).

Karena energi yang sedikit, bisa jadi Mr. Joe melakukan kesalahan dalam tugasnya, misalnya Mr. Joe lupa untuk mengisi tinta yang sudah habis, atau memonya terlalu panjang. Untungnya, Mr. Joe memiliki penolong yang bernama Exception yang dapat menangani kesalahan-kesalahan tersebut. Dengan bantuan Exception, Mr. Joe dapat menghindari kesalahan-kesalahan fatal dalam melaksanakan tugasnya.

Cerita di atas tidak terlalu berhubungan dengan soal ini, jadi tidak perlu Anda baca.

Berikut adalah file header yang diberikan:

- [exception.h](#)
- [memo.h](#)
- [sekretaris.h](#)

Anda perlu membuat dan mengumpulkan **memo.cpp** dan **sekretaris.cpp** sesuai spesifikasi yang diberikan, di mana kelas sekretaris merupakan sekretaris seperti Mr. Joe dan memo merupakan kertas memo yang dapat ditulisi pesan.

Contoh output:

- Jika kertas habis:
"Kertas habis, segera isi kertas" (tanpa kutip)
"Kertas habis" adalah isi pesan dari exception
- Jika berhasil membuat memo untuk Joe:
"Memo [1] untuk Joe berhasil dibuat" (tanpa kutip)
- Setelah membuat memo untuk Joe dan untuk Ban, isi status:
"
Status
Energi : 8
Tinta : 90
Kertas : 3
Memo : 2
Memo [1]
Pesan : Hello
Untuk : Joe
Memo [2]
Pesan : World
Untuk : Ban
" (tanpa kutip)

Catatan:

Gunakan `std::string.length()` untuk mendapatkan panjang string

C++14

 [sekretaris.zip](#)

Score: 190

Blackbox

Score: 190

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	10	Accepted	0.00 sec, 3.05 MB
2	10	Accepted	0.00 sec, 2.93 MB
3	10	Accepted	0.01 sec, 2.93 MB
4	10	Accepted	0.00 sec, 2.99 MB
5	10	Accepted	0.00 sec, 2.88 MB
6	10	Accepted	0.00 sec, 2.89 MB
7	10	Accepted	0.00 sec, 2.98 MB
8	10	Accepted	0.00 sec, 2.98 MB
9	10	Accepted	0.00 sec, 2.89 MB
10	10	Accepted	0.00 sec, 2.89 MB
11	10	Accepted	0.00 sec, 2.99 MB
12	10	Accepted	0.00 sec, 2.93 MB
13	10	Accepted	0.00 sec, 2.90 MB
14	10	Accepted	0.00 sec, 2.96 MB
15	10	Accepted	0.00 sec, 2.99 MB
16	10	Accepted	0.00 sec, 2.95 MB
17	10	Accepted	0.00 sec, 2.95 MB
18	10	Accepted	0.01 sec, 2.91 MB
19	10	Accepted	0.00 sec, 2.96 MB

Time limit	1 s
Memory limit	64 MB

Kariya Ramen adalah sebuah restoran ramen milik Kariya-san. Seorang engineer dari restoran lain yang baru saja pensiun menyarankan Kariya-san untuk menghubungi anda untuk menyelesaikan masalahnya.

Terdapat banyak sekali meja di restoran Kariya Ramen sehingga Kariya-san kesulitan untuk mencatat berapa total yang dihabiskan oleh sebuah meja. Kariya-san meminta anda untuk membuat program yang dapat membantu untuk mengatasi persoalan pada restoran tersebut.

Implementasikan kelas RestoranRamen pada header file [RestoranRamen.hpp](#) yang diberikan berikut ini dengan menggunakan STL map

Berikut merupakan penjelasan dari STL map

Map merupakan sebuah tipe data yang menyimpan nilai key dan value secara berpasangan. Jika didapatkan nilai key maka dapat diambil nilai value. Key pada bersifat unik.

Pada STL map didefinisikan juga sebuah tipe data **pair** yang menggambarkan pasangan nilai (a,b). Terdapat dua pointer atribut pada pair, yaitu first dan second untuk mendapatkan nilai a dan b.

```
#include<map>
#include<iostream>
using namespace std;

// Membuat sebuah pair dengan nilai (1,2)
pair<int,int> sample_pair(1,2);

cout << "Hasil:" << sample.first << endl; // Hasil:1
cout << "Hasil:" << sample.second << endl; // Hasil:2
```

Berikut merupakan beberapa method dasar pada **map**

```
#include<map>
#include<iostream>
using namespace std;

// Membuat sebuah map kosong dengan key bertipe int dan value bertipe int
map<int,int> test_map;

test_map.insert(pair<int,int>(1,5)); // Memasukkan pasangan nilai key 1 dan value 5
test_map.insert(pair<int,int>(2,6)); // Memasukkan pasangan nilai key 2 dan value 6
test_map.insert(pair<int,int>(3,7)); // Memasukkan pasangan nilai key 3 dan value 7

cout << test_map[1] << endl; // Mengambil nilai dengan key 1 (nilai key 1 = 5)
test_map[2] = 3; // Mengubah nilai dengan key 2 menjadi 3
test_map.erase(3); // Menghapus entry map dengan key 3

cout << test_map.size() << endl; // Mengembalikan jumlah entry pada map (jumlah entry tersisa = 2)
```

Seperti halnya STL lain, juga terdapat iterator pada STL map. Akibatnya terdapat juga fungsi seperti **begin()**, **end()**, dan **find(T key)** untuk mengembalikan iterator. Iterator akan mengembalikan `pair<A,B>*` sehingga untuk mengakses key dan value digunakan ->

```
#include<map>
#include<iostream>
using namespace std;

// Membuat sebuah map kosong dengan key bertipe int dan value bertipe int
map<int,int> test_map;

test_map.insert(pair<int,int>(1,5));
test_map.insert(pair<int,int>(2,6));
test_map.insert(pair<int,int>(3,7));

map<int,int>::iterator itr = test_map.begin(); // Mengambil iterator untuk elemen pertama pada map
cout << itr->first << "," << itr->second << endl; // Mencetak 1,5

itr = test_map.find(2); // Mencari data dengan key 2
if(itr != test_map.end()){ // Jika tidak ditemukan .find() akan mengembalikan iterator pada .end()
    cout << itr->first << "," << itr->second << endl; // Mencetak 2,6
} else {
    cout << "Tidak ketemu" << endl;
}
```

Kumpulkan **RestoranRamen.cpp**

C++14

 [RestoranRamen.cpp](#)

Score: 60

Blackbox

Score: 60

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	10	Accepted	0.00 sec, 2.89 MB
2	10	Accepted	0.00 sec, 2.86 MB
3	10	Accepted	0.00 sec, 2.90 MB
4	10	Accepted	0.00 sec, 2.86 MB
5	10	Accepted	0.00 sec, 2.89 MB
6	10	Accepted	0.00 sec, 2.88 MB

[← Latihan Praktikum 2](#)

Jump to...